NUI Galway
OÉ Gaillimh

Semester I Examinations 2010/ 2011

Exam Code(s)	4IF, 4BP, 3BA, 1SD
Exam(s)	4th Year B.Sc. (Information Technology) 4th Year B.E. (Electronic & Computer Engineering) 3rd Year B.A. (Information Technology) Higher Diploma in Applied Science (Software Design & Development)
Module Code(s)	CT404, CT336
Module(s)	GRAPHICS AND IMAGE PROCESSING
Paper No.	I
Repeat Paper	
External Examiner(s)	Prof. M. O'Boyle
Internal Examiner(s)	Dr. J. Duggan * Dr. S. Redfern

Instructions:	Answer any three questions. All questions carry equal marks.
Duration	2 hours
No. of Pages	6
Department(s)	Information Technology
Course Co-ordinator(s)	

Requirements:

MCQ	Release to Library:	Yes	☐	No ☐
Handout				
Statistical/ Log Tables				
Cambridge Tables				
Graph Paper	Required			
Log Graph Paper				
Other Materials				

Q.1.

Consider the OpenGL program below, which draws a 2-dimensional star shape from two triangles (see Fig. 1).

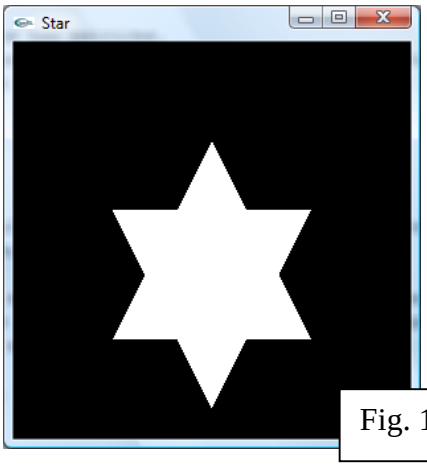

Fig. 1

```
#include <gl/glut.h>

void drawStar() {
  glBegin(GL_TRIANGLES);
    glVertex2f(-0.5, -0.5);
    glVertex2f(0.5, -0.5);
    glVertex2f(0.0, 0.5);

    glVertex2f(-0.5, 0.15);
    glVertex2f(0.5, 0.15);
    glVertex2f(0.0, -0.85);
  glEnd();
}

void display(void) {
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluOrtho2D(-1, 1, -1, 1);

  drawStar();

  glFlush ();
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutCreateWindow ("Star");
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Fig. 2

(a) Modify the display() function so that a 3-dimensional (perspective) projection is used rather than an orthographic projection (*2 marks*)

(b) Modify the display() and drawStar() functions so that the star is drawn at a 3-dimensional location and colour as specified in its parameter list (*6 marks*): `drawStar(float x, float y, float z, float r, float g, float b)`

(c) Extend the program so that a 'starfield' animation is displayed using the Idle Callback approach. Multiple stars should be stored in an array, and initialised to random x, y and z co-ordinates. They should move towards the camera along the z axis, and when they pass the camera they should be moved back to the far distance again. The final program should produce a result similar to that shown in Fig. 2. (*12 marks*)

Note that some useful OpenGL functions, and other standard general-purpose C functions, are listed at the end of this exam paper.

Q.2.

(a) What are *surface normals*? Why are they essential to surface shading and hidden surface removal algorithms? Use diagrams to illustrate your answer. (*5 marks*).

Discuss, with the use of a diagram in each case, the following classic surface shading techniques:

i. Flat (Lambert) Shading (*3 marks*)
ii. Smooth (Gourard) Shading (*3 marks*)
iii. Normal Interpolating (Phong) Shading (*3 marks*)

(b) Consider the X3D code below, which defines a simple scene containing a number of primitive shapes without materials (see Fig. 3).

i. Modify the code so that the cube has a diffuse red colour, the sphere a diffuse green colour, the cylinder a diffuse blue colour, and the cone a diffuse white colour (see Fig. 4) (*3 marks*).
ii. Further modify the code so that a Point Light positioned at the centre of the scene casts a blue light. (See Fig. 5, which shows the result with the 'head light' turned on, and Fig.6 in which the 'head light' is turned off) (*3 marks*)

```
<Scene>
   <Transform translation='-3.0 0.0 0.0'>
      <Shape>
         <Box/>
      </Shape>
   </Transform>
   <Transform translation='0.0 2.0 0.0'>
      <Shape>
         <Sphere/>
      </Shape>
   </Transform>
   <Transform translation='0.0 -2.0 0.0'>
      <Shape>
         <Cylinder/>
      </Shape>
   </Transform>
   <Transform translation='3.0 0.0 0.0'>
      <Shape>
         <Cone/>
      </Shape>
   </Transform>
</Scene>
```

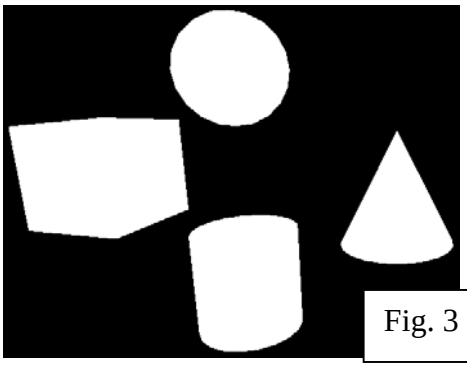

Fig. 3

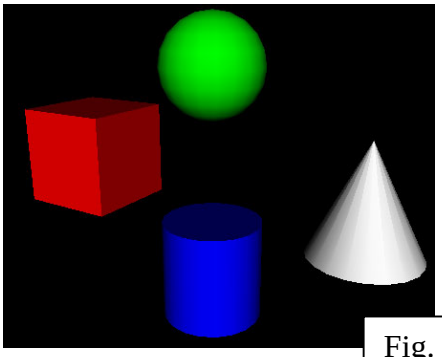

Fig. 4

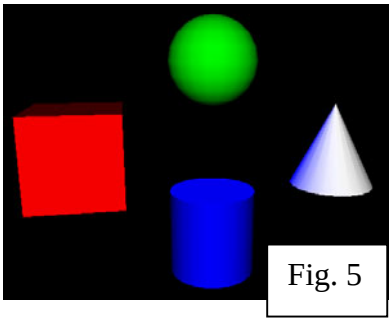

Fig. 5

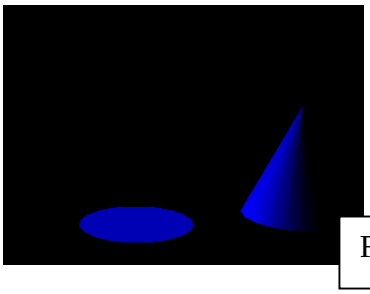

Fig. 6

Note that some useful X3D Nodes are listed at the end of this exam paper.

Q.3.

(a) Describe the use of extrusion in X3D, referring to each of the seven fields used by the Extrusion node. Note that extrusion and other useful nodes from the X3D language are summarised on the final page of this exam paper. (*5 marks*).

(b) Write X3D code to make a model of a free-standing room lamp (example illustrated on the right, Fig. 7).

- Include a diagram of your model showing its measurements, and make the model as geometrically accurate as possible (*9 marks*)
- The lamp should be constructed from a shiny white material (*3 marks*)
- A TouchSensor should be used to turn a PointLight inside the head of the lamp on and off (*3 marks*)

Fig. 7

Q.4.

(a) Describe the morphological image processing techniques of erosion and dilation. Compare the four operations (i) opening, (ii) closing, (iii) thinning and (iv) thickening. In what circumstances might each of these four operations be used? *(10 marks).*

(b) Consider the image of a bubble, shown in Fig. 8.

The image contains substantial amounts of noise, and there exists a large section of bright 'shine' pixels across the centre of the bubble. Propose and justify a series of image processing steps that would be suitable to accurately measure the number of pixels inside the bubble. (*10 marks*).

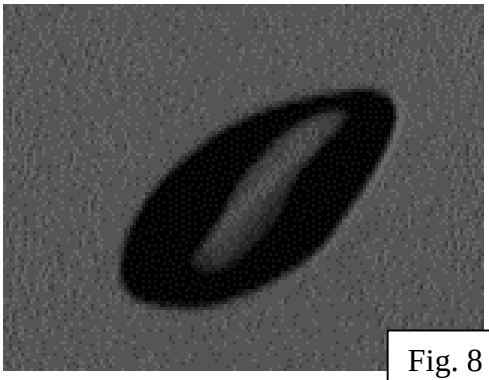

Fig. 8

Q.5.

(a) Transformations in OpenGL are combined using the post-multiplication of matrices. Explain, using a diagram, why the order in which transformations are combined is critical to the resulting transformation. (*5 marks*).

(b) The program below uses wire spheres to display a 'sun' and a 'planet' in OpenGL (see Fig. 9). Using idle-callback animation, modify this program so that the planet rotates around the sun. The planet should rotate around its own centre as well as independently rotating around the centre of the sun. One complete rotation around the sun should take 365 times as long as one rotation around its own centre. (*10 marks*).

```
#include <GL/glut.h>

void display(void) {
   glClear (GL_COLOR_BUFFER_BIT);

   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   gluPerspective(60.0, 1.0, 1.0, 20.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

   glutWireSphere(1.0, 20, 16);   // draw sun
   glTranslatef (2.0, 0.0, 0.0);
   glutWireSphere(0.2, 10, 8);    // draw planet

   glutSwapBuffers();
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("Planets");
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

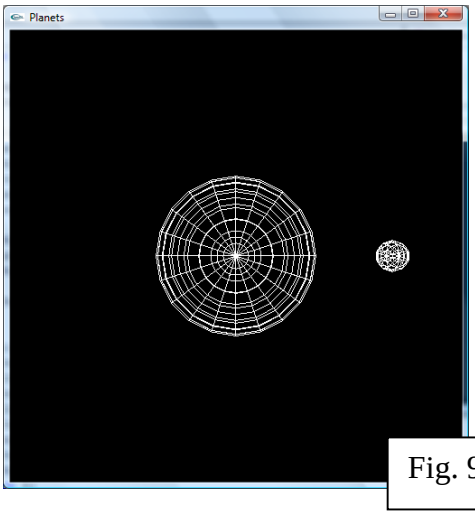

Fig. 9

(c) Explain in general the keyframe approach to animation in computer graphics, and explain specifically its use in X3D, referring to the TimeSensor, Transform, OrientationInterpolator and PositionInterpolator nodes in your answer. (*5 marks*)

Some useful X3D nodes:

Node	Important Fields and Nested Nodes
Shape	Nested Nodes: Appearance, Geometry Nodes (Box, Sphere, Cone, Cylinder, Text, Extrusion, etc.)
Appearance	Nested Nodes: Material, ImageTexture
Material	Fields: diffuseColor, specularColor, emissiveColor, ambientIntensity, transparency, shininess
ImageTexture	Fields: url
Transform	Fields: translation, rotation, scale, center. Nested Nodes: Other Transforms, Shapes, Sensors
TimeSensor	Fields: enabled, startTime, stopTime, cycleInterval, loop
PositionInterpolator	Fields: key, keyValue
OrientationInterpolator	Fields: key, keyValue
Extrusion	Fields: crossSection, spine, scale, orientation, beginCap, endCap, creaseAngle
Box	Fields: size
Sphere	Fields: radius
Cylinder	Fields: radius, height, side, top, bottom
Cone	Fields: height, bottomRadius, side, bottom
PointLight	Fields: on, location, radius, intensity, ambientIntensity, color, attenuation
ROUTE	Fields: fromNode, fromField, toNode, toField

Some useful OpenGL and related functions:

Function	Comments
glBegin(type), glEnd()	Together these delimit a set of vertices. 'Type' defines how to interpret them (GL_POLYGON, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS)
glVertex2f(x,y), glVertex3f(x,y,z)	Define a vertex
glColor3f(r,g,b)	Define the current colour state of OpenGL (red, green, blue)
glMatrixMode(matrix)	Apply subsequent transformations to the specified matrix (GL_MODELVIEW or GL_PROJECTION)
glLoadIdentity()	Remove any existing transforms on the current matrix
gluOrtho2D(left, right, bottom, top)	Define a 2D orthographic projection with specified world coordinate viewport
gluLookAt(eyex,eyey,eyez, atx,aty,atz, upx,upy,upz)	Position the camera (*eye* coords), target it (*at* coords) and define its 'up' vector (*up* coords)
gluPerspective(fov, aspect, near, far)	Define a 3D perspective projection with field-of-view (fov), aspect ratio, near and far clipping planes
glTranslatef(x,y,z)	Translate the coordinate system of the current matrix
glRotatef(a,x,y,z)	Rotate the coordinate system of the current matrix
glScalef(x,y,z)	Scale the coordinate system of the current matrix along each of its principal axes
glutDisplayFunc(display)	Register your display (render) function with glut
glutIdleFunc(idle)	Register your idle function with glut
(float)rand()/RAND_MAX	Produces a random float between 0.0 and 1.0